MOVIE RATING ESTIMATION AND RECOMMENDATION

Zhouxiao Bao, Haiying Xia

ABSTRACT

In this paper, we build a movie rating prediction system based on selected training sets provided by MovieLens. Several machine learning related algorithms – baseline predictor, KNN, Stochastic Gradient Descent, SVD, SVD++, asymmetric SVD, integrated model and NMTF are used to predict the rating from particular users for unrated movies. RMSE (Root-Mean-Square-Error) is applied as the main criteria to evaluate their performance. The simulation result shows distinct performance due to selected algorithms as well as the corresponding learning rates.

Index Terms— Movie rating predictor, SVD, RMSE, Matrix Factorization

1. INTRODUCTION

Recommender systems provide users with personalized suggestions for products or services. They are becoming more and more important in the success of electronic commerce, and being utilized in various applications such as Amazon, YouTube and Google news. Generally speaking, a recommendation system builds up items' profiles, and users' profiles based on their previous behavior recorded. Then it makes a prediction on the rating given by certain user on certain item which he/she has not yet evaluated. Based on the prediction, the system makes recommendations. Various techniques for recommendation generation have been proposed and successfully deployed in commercial environments, among which collaborative filtering (CF) and content-based methods are most commonly used [1, 2].

Movie is now an indispensable entertainment in human life. Most video websites such as YouTube and Hulu and a number of social networks allow users rate on videos/movies. In this project, we build a movie rating prediction system based on several selected training sets, which estimates the movie ratings from each user. According to this result, we are able to make personalized movie recommendations for every user, which would more likely satisfy users' expectation and improve user experience.

2. DATASET AND PRE-PROCESSING

Considering about the simulation efficiency, we choose MovieLens 100k Data Set1 as our training and testing set. Such set consists of 100,000 ratings (rating score from 1-5) from 943 users on 1682 movies, in which each user has rated at least 20 movies. Among MovieLens 100k Data Set, we choose ua.base/test and ub.base/test for result comparison between different training sets and testing sets; and ua.base/test for model comparison between different algorithms. The whole set u data is split into a training set and a test set with exactly 10 ratings per user in the test set. The sets ua.test and ub.test are disjoint. Also, in order to apply cross validation to select from different models, we again split the data set ua.base into 10 small disjoint sets. This allows us to perform 10-fold cross validation on ua1.base - ua10.base and evaluate the finally selected model on test set ua.test.

The training and testing data are pre-processed as follows: we apply the database to build a U by I matrix A, where U is the number of users, and I is the number of rated movies. Each element A_{ui}denotes the rating scored by the u-th user for the i-th movie. It is easy to find that the majority of movies don't obtain a sufficient number of ratings, and also, there only exist common ratings for general user. So A is a very sparse matrix.

3. PROBLEM DESCRIPTION

After pre-processing, we obtain a large user-item matrix $A \in \mathbb{R}^{N_u \times N_i}$, where N_u is the number of users and N_i is the total number of items (movies). So we have:

$$A_{u,i} = \begin{cases} R_{ui}, & \text{existent rating} \\ 0, & \text{no such rating} \end{cases}$$

Thus our work is to fill in all the zero-entries in the matrix based on the training set of existing ratings. Assume the prediction rating of user u on item i is $T_{ui}(R_{ui} = 0)$.

In this project the widely used RMSE (Root-Mean-Square Error) criteria is applied to evaluate the performance of each algorithm, it could be calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{|S_{test}|}\sum_{(u,i)\in S_{test}} (R_{ui} - T_{ui})^2}$$

As is mentioned above, $(u, i) \in S_{test}$ actually means that user u has rated item i before.

4. ALGORITHMS

4.1 Baseline predictors

We use a simple baseline predictor to estimate the movie ratings from each particular user. The predictor algorithm is described in [2]. In this approach, the unknown rating score is estimated as $b_{ui} = \mu + b_u + b_i$, where μ is the overall average score, b_u and b_i is the training deviations of user u and item i, respectively. The parameters, b_u and b_i are estimated using a decoupling method, which requires less complexity but less accurate. We choose $\lambda_2 = 25, \lambda_3 = 10$, which are parameters applied to estimate b_u and b_i , respectively.

By applying ua.base, ub.base as two training sets, and ua.test, ub.test as two test sets for them, respectively.

Running the baseline predictor, we get the result listed in Table 1.

Table 1: Performance of Baseline

	ua Data	ub Data
RMSE	0.96648	0.97737

It shows that by choosing proper parameters λ_2, λ_3, we can obtain reasonable results on rating prediction. Note that different λ_2, λ_3's will result in distinct RMSE. Our goal here is just to provide some values to compare with following results, so only one pair of reasonable parameters is used here.

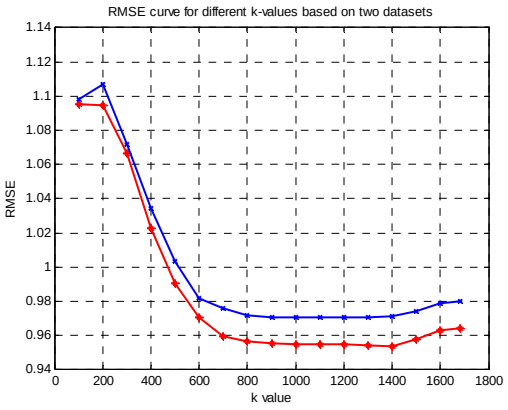
4.2 KNN (K-Nearest Neighbor)

In the problem that we apply KNN to estimate the movie rates, we use a similarity matrix to measure the "distance" between each item. So, this method should be more precisely named as item-based KNN algorithm. The approach is described in detail in [3]. To measure the similarity between items, we choose Pearson correlation. The similarity between item i_1 and item i_2 can be calculated as follows:

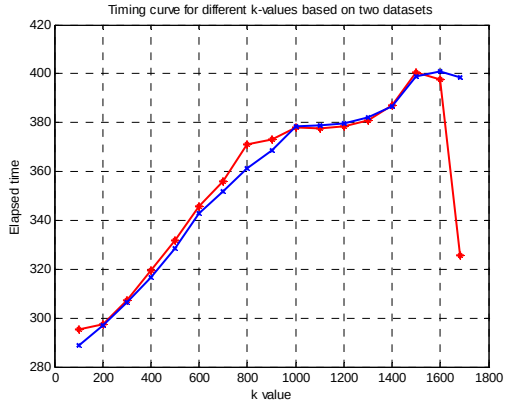
$$\text{Similarity}(i_1, i_2) = \frac{\sum_{u \in U(i_1) \cap U(i_2)} (R_{u,i_1} - \bar{R}_u)(R_{u,i_2} - \bar{R}_u)}{\sqrt{\left(\sum_{u \in U(i_1) \cap U(i_2)} (R_{u,i_1} - \bar{R}_u)^2\right)\left(\sum_{u \in U(i_1) \cap U(i_2)} (R_{u,i_2} - \bar{R}_u)^2\right)}}$$

Here, $U(i_1)$ is the set of all users who has rated on i_1 before; $U(i_2)$ is the set of all users who has rated on i_2; $U(i_1) \cap U(i_2)$ is a set of users who has rated both items; $\bar{R}_u$ is the average rating of all ratings given by user u.

Since the performance depends on k, number of nearest neighbors, we try different k's in the same training set, and evaluate the corresponding performance by measuring the RMSE. Using two database ua and ub, the RMSE curve based on different k-values is shown in Fig.1(a), while the corresponding timing curve is illustrated in Fig.1(b).



(a)



(b)

Fig. 1 The (a) RMSE curve and (b) timing curve based on different k-values.

From the above curves, we are able to choose a proper k-value to achieve reasonable estimation performance. For the particular training set in our project, an appropriate k-value is suggested ranging from 800 to 1300.

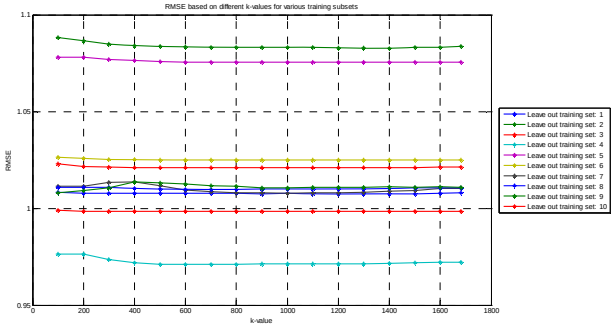


Fig. 2 RMSE curve based on different k-values for leaving out different training subsets.

The above model is then further improved by using 10-fold cross validation. We split the data set ua.base into 10 sets and perform 10-fold cross validation on ua1.base - ua10.base. The RMSE curve in terms of different k-values on the 10 training subsets is illustrated in Fig.2. The

suggested k-value according to the curves is ranging from 900 to 1000, which is similar to the result without cross validation. The smallest RMSE value in the overall simulation result is 0.95534.

4.3 Stochastic Gradient Descent

The rating estimation equation for Stochastic Gradient Descent method could be written as:

$$\hat{r}_{ui} = q_i^T p_u,$$

where $q_i \in \boldsymbol{R}^k$ describes the overall interests of the users to particular items, and $p_u \in \boldsymbol{R}^k$ illustrates the interests one particular user has for the items. Denote the prediction error $e_{ui} = r_{ui} - \hat{r}_{ui}$, then q_i, p_u are updated using gradient descent following the equations:

$$q_i \coloneqq q_i + \alpha(e_{ui} \cdot p_u - \beta \cdot q_i)$$
$$p_u \coloneqq p_u + \alpha(e_{ui} \cdot q_i - \beta \cdot p_u)$$

where α, β are learning rates, and the initial values for q_i, p_u are set as random vectors with entries uniformly distributed in [0,1]. After several experiments for different choice of α, β, we select $\alpha = 0.01, \beta = 0.05$, the RMSE curve with different k values is shown in Fig. 3.

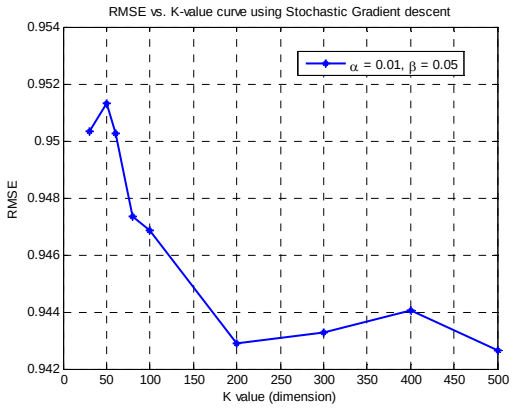


Fig. 3 RMSE curve based on different k-values(dimension) using Stochastic Gradient Descent method

The RMSE curve shows a better performance than both baseline predictor and KNN.

4.4 SVD and its variants

4.4.1 SVD

The SVD method could be considered as combination of baseline predictor and Stochastic Gradient Descent. The overall prediction equation is written as in [2]:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

with μ, b_i, b_u, q_i, p_u defined in section 4.1 and 4.3. In our project, b_i, b_u are initialized using decoupling method, and q_i, p_u 's initial values remain to random vectors. Their updating rules are:

$$b_u \coloneqq b_u + \alpha_1 \cdot (e_{ui} - \beta_1 \cdot b_u)$$
$$b_i \coloneqq b_i + \alpha_1 \cdot (e_{ui} - \beta_1 \cdot b_i)$$
$$q_i \coloneqq q_i + \alpha_2 \cdot (e_{ui} \cdot p_u - \beta_2 \cdot q_i)$$
$$p_u \coloneqq p_u + \alpha_2 \cdot (e_{ui} \cdot q_i - \beta_2 \cdot p_u)$$

In our project, we choose the learning rate $\alpha_{1,2} = 0.01$, $\beta_{1,2} = 0.05$ for SVD method.

4.4.2 SVD++

Based on SVD method, SVD++ improves the accuracy by adding an item related factor vector $y_i \in \boldsymbol{R}^k$. As described in [2], the prediction model becomes:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T (p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j)$$

where $R(u)$ is defined as the set in which the corresponding items are rated by user u. The updating equation is also improved as provided in [2]. In our project, the learning rate is chosen as $\alpha_1 = 0.009, \alpha_2 = 0.005, \beta_1 = 0.045, \beta_2 = 0.05$ for a relatively good performance.

4.4.3 Asymmetric SVD

The asymmetric SVD method improves the base SVD as described in [6]. In our project, we revise the estimation equation as in order to reduce computing complexity:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \cdot |R(u)|^{-\frac{1}{2}} \cdot \sum_{j \in R(u)} [(r_{uj} - b_{uj}) x_j + y_j]$$

A relatively better learning rate set is chosen as: $\alpha_{1,2} = 0.001$, $\beta_{1,2} = 0.01$.

The RMSE curves based on different k-values (dimension) for SVD and its variants are in Fig. 4.

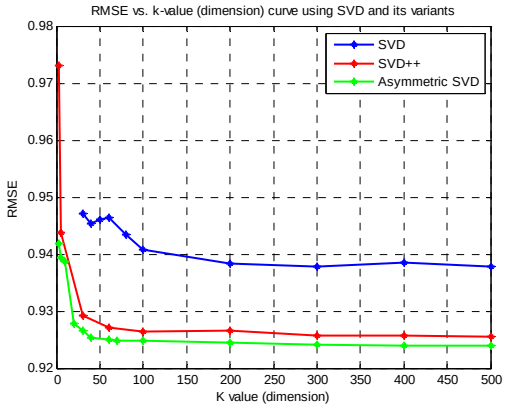


Fig. 4 RMSE curve based on different k-values(dimension) using SVD and its variants

4.5 Global Neighborhood model

This model allows an efficient global optimization scheme. It is able to integrate implicit user feedback. We abandon the user-specific weights in favor of global weights which are independent of a specific user. The estimation is:

$$\hat{r}_{ui} = \mu + b_u + b_i + |R(u)|^{-1/2} \sum_{j \in R(u)} \left[(r_{uj} - b_{uj}) w_{ij} + c_{ij} \right]$$

The updating rule for w_{ij} and c_{ij} is:

$$w_{ij} \leftarrow w_{ij} + \gamma \left(|R(u)|^{-\frac{1}{2}} e_{ui} (r_{uj} - b_{uj}) - \lambda w_{ij} \right)$$
$$c_{ij} \leftarrow c_{ij} + \gamma \left(|R(u)|^{-\frac{1}{2}} e_{ui} - \lambda c_{ij} \right)$$

Apply this algorithm on ua and set the parameters as: $\gamma = 0.007, \lambda = 0.019$, we can get for uatest is:

RMSE = 0.931926

This is much better than the baseline and KNN predictor, but is worse than those matrix factorization methods.

4.6 Co-clustering by BVD (Block Value Decomposition)

Recently, traditional data clustering methods such as K-means has been applied in the transformed space. Different from SVD, it introduces a block value matrix. The algorithm is given by the minimization of:

$$f(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{M}) = \|\boldsymbol{A} - \boldsymbol{UBM}\|^2$$

where $\mathbf{U} \in \mathbb{R}^{\mathbf{N_u} \times \mathbf{k}}$ is user-cluster matrix, $\mathbf{M} \in \mathbb{R}^{\mathrm{k} \times \mathrm{N_i}}$ is movie-cluster matrix and $\mathbf{B} \in \mathbb{R}^{\mathrm{k} \times \mathrm{k}}$ is block value matrix (k is the number of clusters we select).

Apparently, the objective function is convex in U, R and M. So we can derive an EM style algorithm that converges to a local minimum by iteratively updating the decomposition using a set of multiplicative updating rules:

$$\mathbf{U}_{ij} \leftarrow \mathbf{U}_{ij} \frac{(\mathbf{A}\mathbf{M}^T\mathbf{B}^T)_{ij}}{(\mathbf{U}\mathbf{B}\mathbf{M}\mathbf{M}^T\mathbf{B}^T)_{ij}}$$
$$\mathbf{B}_{ij} \leftarrow \mathbf{B}_{ij} \frac{(\mathbf{U}^T\mathbf{A}\mathbf{M}^T)_{ij}}{(\mathbf{U}^T\mathbf{U}\mathbf{B}\mathbf{M}\mathbf{M}^T)_{ij}}$$
$$\mathbf{M}_{ij} \leftarrow \mathbf{M}_{ij} \frac{(\mathbf{B}^T\mathbf{U}^T\mathbf{A})_{ij}}{(\mathbf{B}^T\mathbf{U}^T\mathbf{U}\mathbf{B}\mathbf{M})_{ij}}$$

Running this algorithm for different values of k until converge, we can get the following curve of RMSE on dataset ua.

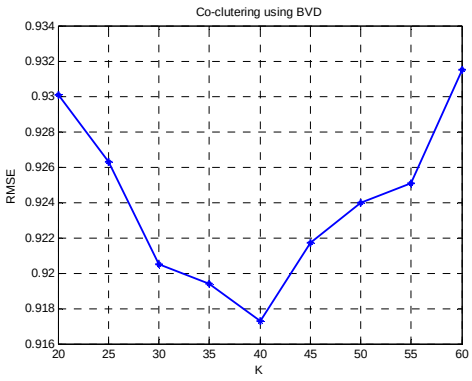


Fig. 5 RMSE vs. k-values(dimension) for Co-clustering using BVD

This algorithm successfully captures the clustering feature of both users and movies and the relationship between their clusters. From the result, we can see that this algorithm can generate better performance than previous methods. It can reach a smallest RMSE at certain K, and this value, we found, depends highly on the characteristic of the dataset.

5. CONCLUSION

In our project, various rating prediction algorithms are applied using the MovieLens dataset. The methods based matrix factorization outperform the other ones, which only use the stochastic information of the training database. Adding the block value matrix can further improve the performance.

6. REFERENCES

[1] J. Breese, D. Heckerman and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering, Technical Report of Microsoft Research, 1998.

[2] F. Ricci, L. Rokach, B. Shapira, P. Kantor, Recommender Systems Handbook.

[3] B. Sarwar, G. Karypis, J. Konstan and John Riedl, Item-Based Collaborative Filtering Recommendation Algorithms, Proceedings of the 10th international conference on World Wide Web 2001: 285-295.

[4] B. Long, Z. Zhang and P. Yu, Co-clustering by Block Value Decomposition, SIGKDD'05, August 21-24, 2005.

[5] T. Huynh and D. Vu, Using Co-clustering for Predicting Movie Rating in Netflix.

[6] Y. Koren, Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, KDD'08.